Fuzzy-Logic Control System

Designing a ferryboat docking system using both conventional control equations and fuzzy logic shows you the strengths and limitations of both technologies.

Doug Conner, Technical Editor

You're designing a control system. If you have complex equations, lots of exceptions to the equations, nonlinearities to accommodate, system inputs that provide vague or ambiguous information—or no equations at all—you might want to abandon conventional control-system design and try using fuzzy logic.

For those who haven't used fuzzy logic, making the transition is filled with the usual concerns of learning and using a new technology while maintaining a schedule. To help you see what learning to use fuzzy logic is like, I worked through a simple control problem using first a conventional approach and then fuzzy logic. (If you haven't read a fuzzy-logic article before, you might want to start with David I Brubaker's "Everything you always wanted to know about fuzzy logic" in this issue.)

I'm not a fuzzy-logic expert. I started at ground zero using Motorola's fuzzy-logic educational kit and tutorial information from several other companies' software tools (**Table 1**). The control problem I tackled was docking a ferryboat. This task is well suited for conventional control systems, and fuzzy logic isn't necessary. I chose this example because the simple physics involved lets you intuitively understand what is happening. The example doesn't showcase fuzzy logic's ability to solve particularly difficult problems, but I hope it provides insight into understanding and applying fuzzy logic.

The control system needs to decelerate a ferryboat to near-zero speed by the time it impacts the energy-absorbing barrier at the dock. Because a ferryboat typically approaches the dock head on and must impact the barrier at very low speed, the control system requires precision. The system is for use with boat speeds from 0 to 4m/sec during the last 200m before reaching the dock and propeller speeds of −1000 to +1000 rpm.

The basic requirements are to reach the dock in a short time, to use relatively smooth deceleration, and to have near-zero velocity when impacting the barrier (less than or equal to 0.1m/sec). The boat's mass is between 500,000 and 1,000,000 kg.

A conventional control approach

I first solved the problem using a conventional control approach. I decided to control the boat on a velocity-versus-distance profile that would require a constant target thrust at maximum gross mass (m). The target distance is the distance you want the boat to be from the dock for any given velocity. Using basic physics, you can derive

the target distance vs velocity (v) equation as

$$\text{target distance} = 0.5mv^2/\text{target thrust}.$$

Subtracting the target distance from the actual distance generates a distance error. The control system uses the distance error to regulate the rpm-command setting for the boat's propeller. To make the system more sensitive to position errors as the boat approaches the dock, I divided the distance error by the distance to the dock (N). When the distance is less than 0.1m, normalization term N remains at 0.1 to avoid making the system too sensitive and avoid division by zero.

The equation for the rpm command is

$$\text{rpm command} = K_{ENGINE} \times \frac{\text{distance error}}{N},$$

where $K_{ENGINE} \times$ is the engine-control gain.

I used these two equations to simulate the control system. **Fig 1** shows the results. The simulation includes approximations for the time the engine needs to respond to rpm-command changes and for thrust as a function of propeller rpm and boat velocity. **Fig 1a** shows a plot of the velocity versus distance for two cases with different boat mass and initial velocity and the target velocity-versus-distance profile. The control system will have a small position error, which will result in a very small velocity when the boat impacts the barrier.

Fig 1b shows a plot of the propeller rpm versus distance from the dock for the same two cases. You can see the control system does a good job of using a near-constant-rpm deceleration after the boat's velocity has approximately merged with the target profile in **Fig 1a**. For a real ferryboat, you'd also want to include a function to provide a smooth transition from the cruise phase into the docking phase.

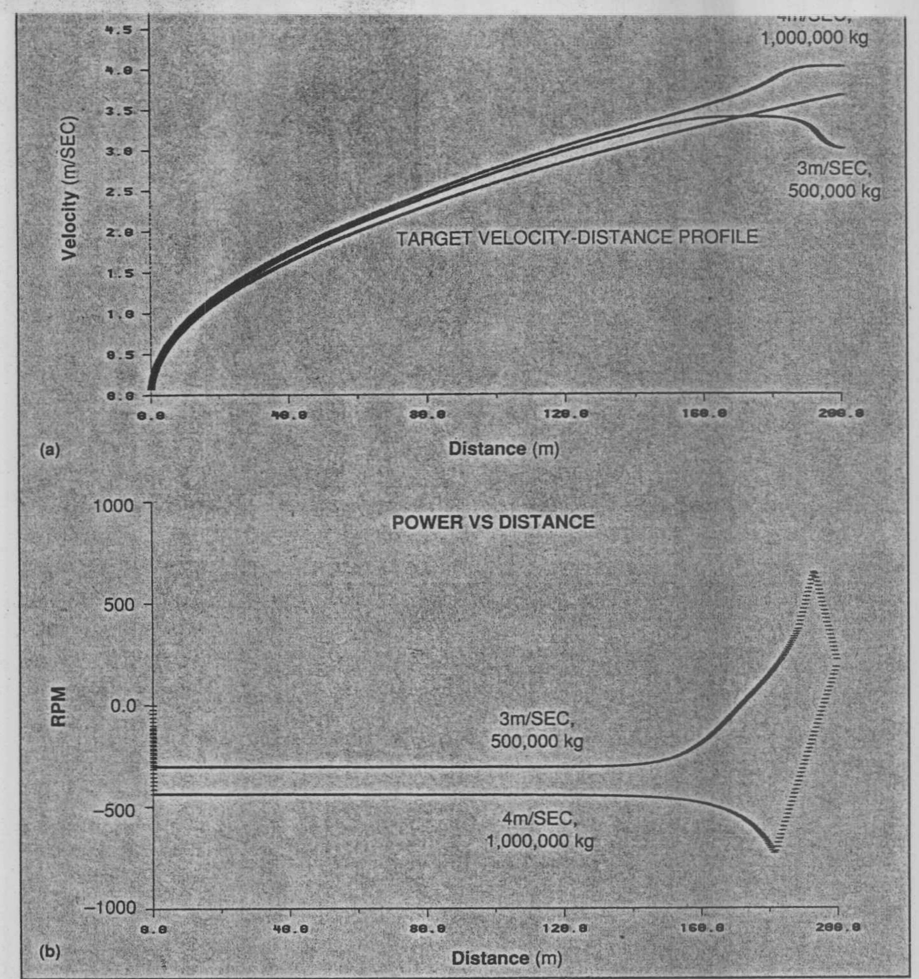

Fig 1—Simulating ferryboat docking using my conventional control system proves that the system works well. Plot (a) shows distance versus velocity for a 1,000,000-kg boat with a 4m/sec initial velocity and a 500,000-kg boat with a 3m/sec initial velocity. It also shows the target, or ideal, velocity-versus-distance profile. The simulation in (b) starts with the propeller at 200 rpm and shows a plot of distance versus propeller rpm for the same two cases. After stabilizing on a velocity-versus-distance profile, the control system does a good job of maintaining a constant propeller rpm.

To get a better understanding of how the system would respond to any combination of velocity and distance inputs, look at the control surface in **Fig 2**. A control surface typically provides 3-D information, with two axes being inputs and the third being the output.

Cubicalc—the software from Hyperlogic I used for both the conventional and fuzzy-logic simulations in this article—provides only 2-D plots, but you can generate plots with 3-D information by using a topographic format. You can see from the control surface that the control system takes the ferryboat starting at some point on the right side of the control surface (0 to 4m/sec velocity and 200m distance) and adjusts propeller rpm to follow a specific velocity-versus-distance profile.

The 0-rpm line in **Fig 2** is the target approach profile. For input-velocity and distance cases above this line, the control surface shows that the system will issue a negative-rpm command to the propeller to slow the boat. For input-velocity and distance cases below the 0-rpm line, the system will issue the propeller a positive-rpm command to accelerate the boat.

Designing a fuzzy-logic control system

Incidentally, I used Cubicalc because the software lets you easily integrate conventional math functions with fuzzy logic and quickly and easily construct simulations. I could make changes to a simulation and be looking at plotted results in less than a minute.

My conventional control system appears to work well for the ferry-boat-docking problem. Now, I approach the problem using fuzzy

Table 1—Fuzzy-logic tools and products

Company	Product	Description	Price
American Neuralogix Inc	NLX 230 fuzzy microcontroller	Has 8 digital inputs, 8 digital outputs, 16 fuzzifiers; holds 64 rules. Evaluates 30M rules/sec.	$9.56 (100)
	ADS 230 fuzzy microcontroller development system	PC-compatible system uses NLX 230 with analog and digital I/O.	$395
	NLX 110 fuzzy pattern correlator	Correlates eight 1-Mbit patterns; expandable to 256 *n*-bit patterns	$23.83 (100)
	NLX 112 fuzzy data correlator	Performs pattern matching on serial data streams.	$20.80 (100)
Aptronix Inc	Fide (Fuzzy Interference Development Environment)	Runs under MS-Windows on 386/486 PCs. Supports development, fuzzy simulation, debug tracing, and 3-D display of control surfaces. Real-time code generation for microcontrollers. Software implementation of fuzzy logic in C. Complete tutorial information and phone support.	$1495 (Motorola is a distributor)
Byte Craft Ltd	Fuzz-C	Preprocessor translates fuzzy source code into C source code.	$149
Fuzzy Systems Engineering	Manifold editor	Runs under MS-Windows 3.1 on 386/486 PCs. Edits rules in a matrix display. Lets you view fuzzy sets graphically.	$295
	Manifold graphics editor	Runs under MS-Windows 3.1 on 386/486 PCs. Color graphics display of rules and fuzzy sets. Lets you view designs in 3-D map and slice formats.	$495
Hitachi America Ltd	Microcontrollers	The company has performance benchmarks for its H8/300 and H8/500 microcontrollers in fuzzy-logic applications, performed by Togai Intralogic.	$5.40 to $17.45 (10,000)
Hyperlogic Corp	Cubicalc	Software for developing fuzzy-logic applications. Runs under MS-Windows with 286 or higher processor. Simulates fuzzy and non-fuzzy systems.	$495
	Cubicalc-RTC	A superset of Cubicalc. Provides runtime compiler support and libraries for linking. Compatible with Microsoft C and Borland C.	$795
	Cubicalc runtime source code	Generates C source code for use in compiling to a specific processor.	$995
	Cubicard	Includes Cubicalc-RTC and PC-based hardware for analog and digital I/O.	$1495
Inform Software Corp	Fuzzytech Explorer Edition	Introductory fuzzy-logic system. Software runs under MS-Windows. Accepts two inputs, one output, and five fuzzy membership sets per variable and 125 rules. Includes tutorial.	$199
	Fuzzytech MCS-96 Edition	Full fuzzy development system for MCS-96 microcontrollers. Generates optimized assembly code.	$1890
	Fuzzytech Online Edition	Lets you debug and modify fuzzy-logic systems while they are running. Generates C source code.	$6900
Integrated Systems Inc	RT/Fuzzy Module	Simulation and code generation of fuzzy logic for real-time systems.	$5000
The Metus Systems Group	Metus	Fuzzy-logic development and simulation system. Runs under MS-DOS. Provides high-level modeling and low-level development for embedded applications.	$1250
Modico Inc	Fuzzle 1.8	PC-based fuzzy-logic shell. Generates source code for C and Fortran.	$289
Motorola Inc	Fuzzy-logic kernel for microcontrollers	Fuzzy processing kernels for 68HC05 and 68HC11 microcontrollers. Includes fuzzy knowledge-base generator to create code for kernel.	Free
	Fuzzy-logic educational kit	Interactive training tool provides good introduction for understanding and using fuzzy logic. Runs under MS-Windows. Includes demonstration version of Fide (from Aptronix).	$195
Togal Infralogic Inc	TILShell+ fuzzy C development system	Complete fuzzy development system generates C code and includes debug, fuzzy-simulation, and graphical-analysis tools. Tutorial included.	$4600 (PCs) $6900 (Sun workstations)
	Microcontroller evaluation packages	Fuzzy development systems for Hitachi H8/300, H8/500, and HMCS400; Intel 8051; and Mitsubishi 37450.	$2500 (per processor)
	Microcontroller production licenses	Unlimited production license.	$9000 (per processor)
	FC110	Digital fuzzy-logic processor (IC).	$40 (100)
	FC110 development system	Hardware and software development system for FC110. Versions support IBM PC/AT bus, Sbus, and VMEbus.	$645 to $4500

logic. I intentionally avoid directly translating the conventional approach into fuzzy logic. I start by trying to solve the problem without using the knowledge I gained from the conventional control equations.

I know from the tutorials I've studied that fuzzy logic doesn't require that I be able to mathematically model the control problem. What I need is an understanding of the problem and an everyday understanding of physics. Here are the basic assumptions I started with:

1. Reverse thrust will decrease the boat's forward velocity, and forward thrust will increase it.
2. The closer the boat is to the dock, the slower it should be going.

The first step in developing a fuzzy-logic control system is determining the system inputs, system outputs, and the fuzzy rules.

For the ferryboat problem, the inputs are distance to the dock and velocity, and the output is the rpm command for the propeller. Now I need to define fuzzy sets for the input and output variables. After some experimenting, I settled on the fuzzy sets in **Fig 3**.

The fuzzy rules

Next, I translate my knowledge of the control problem into fuzzy-logic rules. Here are the rules I settled on after some more experimentation:

1. If Velocity is Positive_Large, then RPM_Command is Negative_Large.
2. If Distance is Positive_Large AND Velocity is Postive_Large, then RPM_Command is Zero.
3. If Distance is Positive_Large AND (Velocity is Postive_Small OR Velocity is Near_Zero), then RPM_Command is Positive_Large.
4. If Distance is Positive_Small AND Velocity is Positive_Large, then RPM_Command is Negative_Large.
5. If Distance is Positive_Small AND Velocity is Positive_Small, then RPM_Command is Zero.
6. If Distance is Positive_Small AND Velocity is Near_Zero, then RPM_Command is Positive_Large.
7. If Distance is Near_Zero AND (Velocity is Positive_Large OR Velocity is Positive_Small), then RPM_Command is Negative_Large.

The rules typically work in opposition—some try to accelerate the boat, and others try to decelerate it.

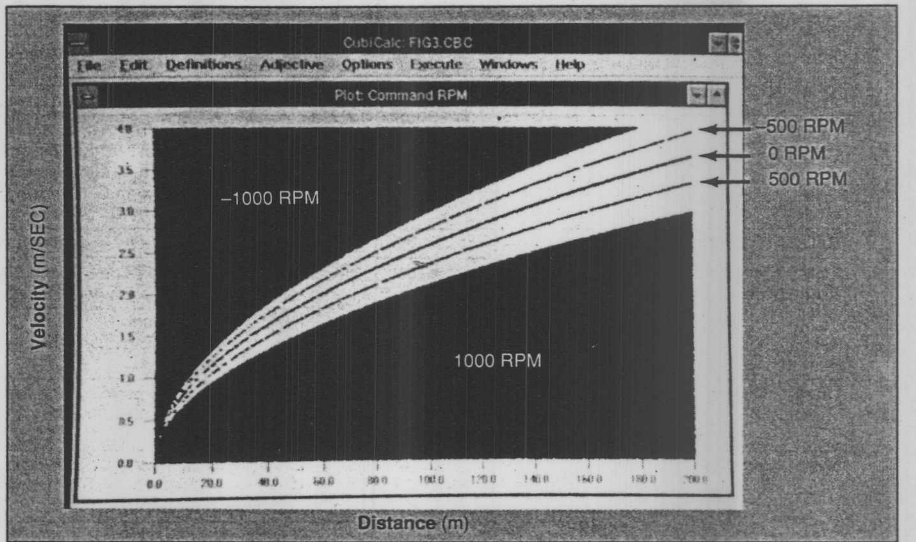

Fig 2—The control surface for a conventional ferryboat-docking control system results from displaying 2-D information in a topographic format. For any combination of distance and velocity inputs, the control surface shows the output rpm command. The thick black lines show constant rpm in ± 20-rpm slices.

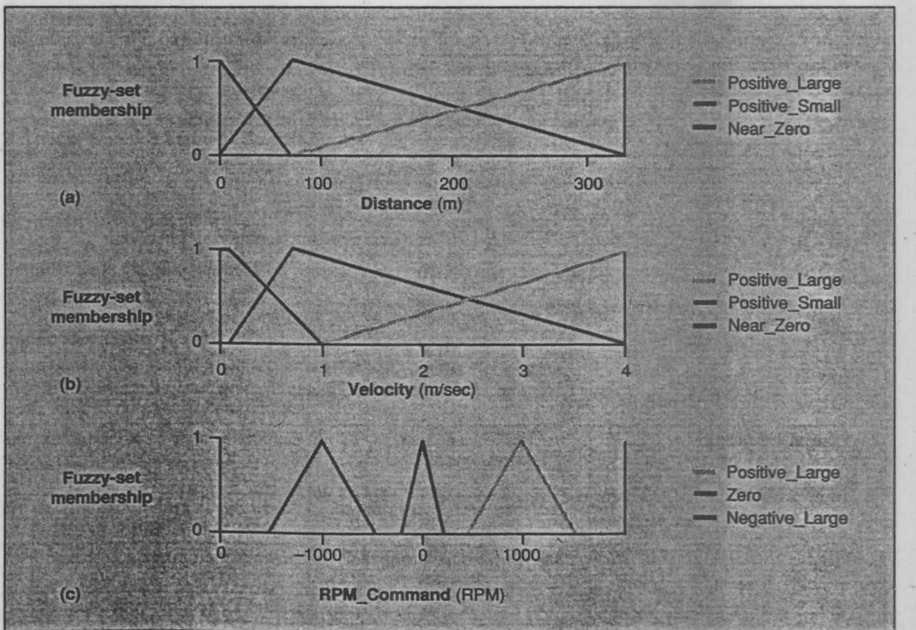

Fig 3—These three plots define the fuzzy sets for the input variables Distance (a) and Velocity (b) and the output variable RPM_Command (c). These plots plus the 7-rule set define my first fuzzy-logic ferryboat-docking control system.

When fuzzy logic evaluates the AND operation, the result is the antecedent (the "if" part of the if-then statement) with the smallest set membership. An OR operation results in the antecedent with the largest set membership. **Fig 4** demonstrates how to evaluate the fuzzy-logic antecedents of rule 4 to arrive at a consequent (the "then" part of the if-then statement) for a particular set of inputs).

Because more than one rule will usually apply for any set of inputs, fuzzy logic must be able to combine the results of several rules. This combining is called defuzzification—the process of getting a "crisp" single-value output from multiple output fuzzy sets. One way to combine results is to take the maximum set-membership values of the active consequents and then perform a center-of-gravity computation to obtain a single output value. The center-of-gravity method is one of the more common defuzzification methods. Other methods are sometimes useful for systems with special requirements such as high computational speed.

Figs 5 and 6 show the performance of my fuzzy-logic control system. **Fig 5a** shows two velocity-versus-distance profiles; **Fig 5b** shows the distance to the dock versus propeller rpm. You can see the curves aren't as smooth as those of the conventional control system. Probably most disturbing to passengers would be the abrupt rpm changes that occur at around 40m (**Fig 5b**).

The control surface in (**Fig 6**) also isn't as smooth as that of the conventional control system (**Fig 2**). When I created the membership sets and rules for the fuzzy-logic system, I had hoped the continuous nature of fuzzy logic would smooth the rough velocity-versus-distance profile described by my seven rules. Although the 0-rpm line is relatively smooth, the other lines show

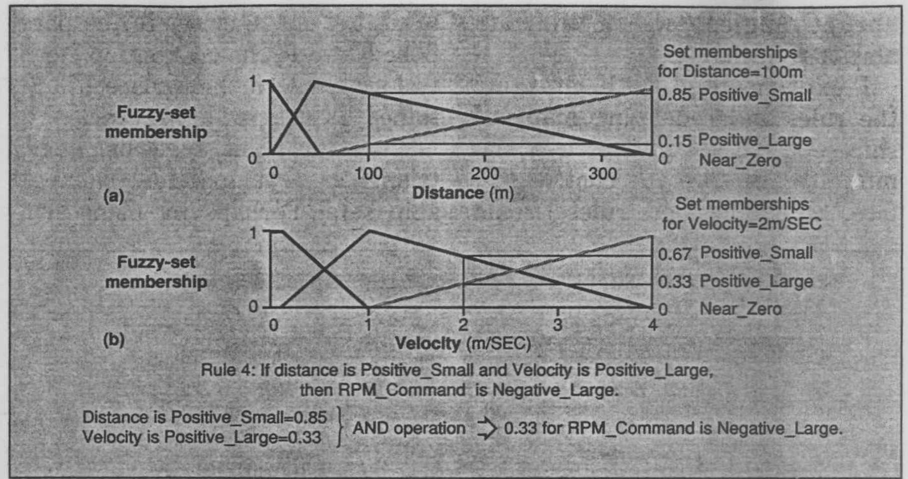

Fig 4—A distance of 100m has a membership value of 0.85 in the Positive_Small fuzzy set for Distance (a). A velocity of 2 m/sec has a membership value of 0.33 in the Positive_Large fuzzy set for Velocity (b). Because rule 4 generates an output by using the AND operation, the output is the minimum of the two inputs: 0.33 Negative_Large for RPM_Command.

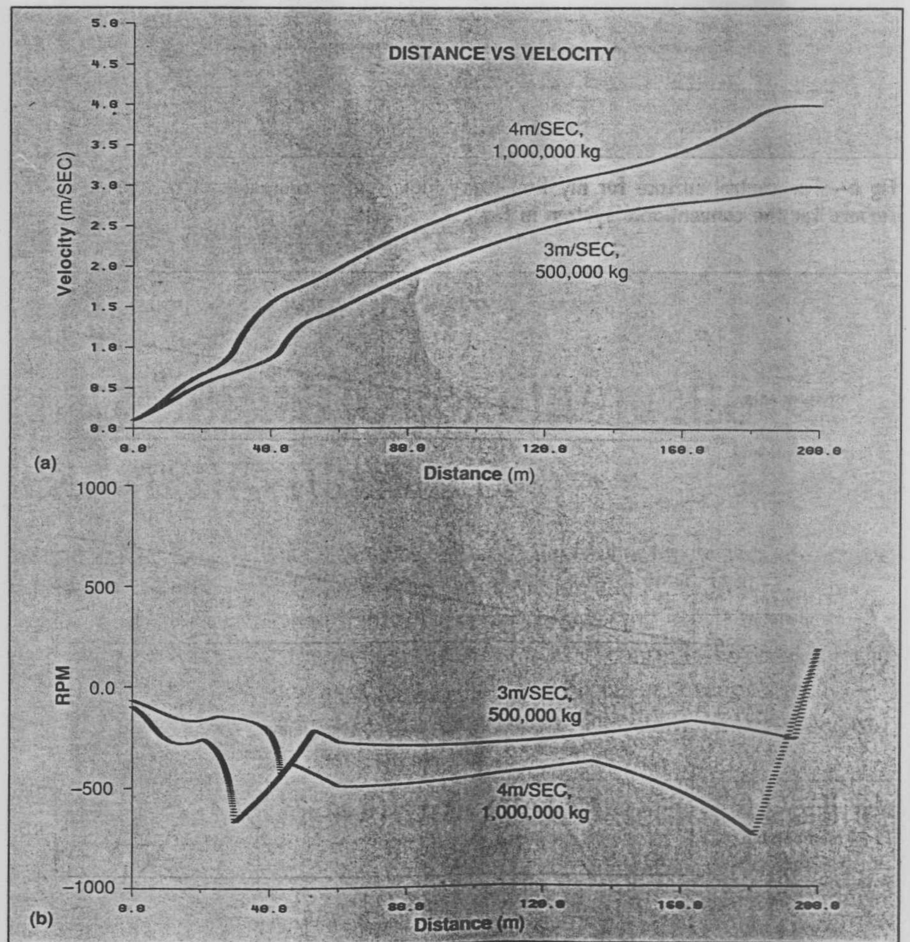

Fig 5—Simulating my first attempt at a fuzzy-logic control system shows that my system is unacceptable. Plot (a) shows the distance versus velocity for the two cases I used in Fig 1a. The distance-versus-rpm plot for the same two cases in (b) shows that the thrust is not constant. The simulation starts with the propeller at 200 rpm.

abrupt angles, which indicate abrupt rpm changes.

I experimented with changing the rules and redefining membership sets, but I wasn't able to significantly improve the control surface. By adding more rules I could bring the shape closer to the parabolic curve of the conventional control system, but the surface still retained its stepped appearance.

Up to this point, I was using only triangular or trapezoidal membership sets. Perhaps by using nonlinear membership sets I could smooth the control surface into the parabolic shape of the conventional control system. But before adding to the system's complexity, I wanted to simplify the fuzzy-logic rules and membership sets as much as possible.

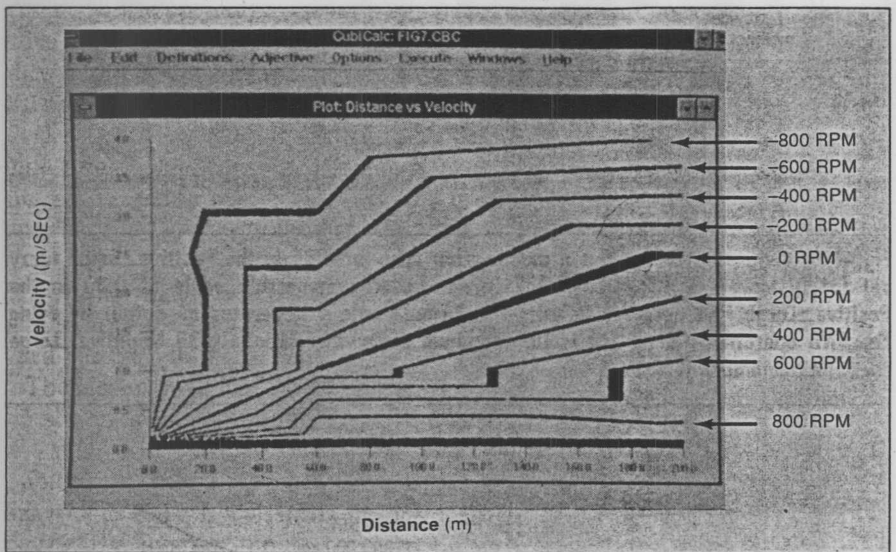

Fig 6—The control surface for my first fuzzy-logic control system isn't as smooth as the surface for the conventional system in Fig 2.

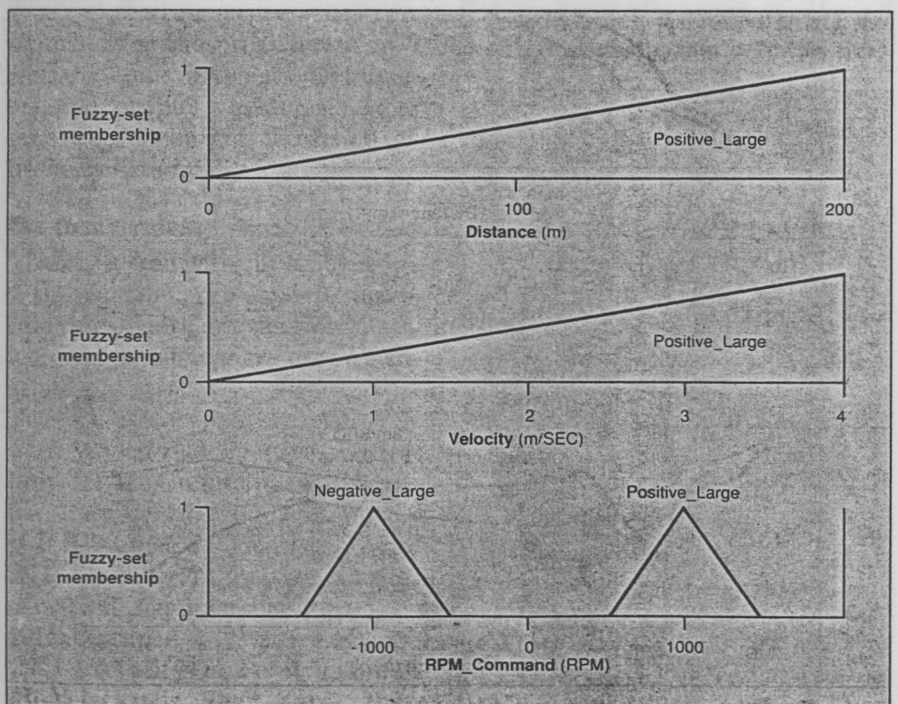

Fig 7—The fuzzy sets for my second attempt at a fuzzy-logic ferryboat-docking system are much simpler than the sets I used for my first try. I reduced the number of membership sets to four and the corresponding rules to two.

A second attempt with fuzzy logic

After some thought and experimentation, I came up with the following rule set; I don't think it could be simplified any further:

1. If Distance is Positive_Large, then RPM_Command is Positive_Large.
2. If Velocity is Positive_Large, then RPM_Command is Negative_Large.

Fig 7 shows the linear membership sets for my new 2-rule set. **Fig 8** shows the control surface for this new fuzzy control system—a series of simple, linear velocity-versus-distance profiles.

By adding some gain, I might be able to make this system work, but it has deficiencies. First, the linear velocity-versus-distance approach to the dock takes a long time. In fact, mathematically the boat would never reach the dock because the boat's velocity approaches 0 as the boat approaches the dock. In practice, it's acceptable for the boat to have some small positive velocity when it reaches the dock, so you could put an offset in the set-membership functions. But even with practical modifications, the time to reach the dock is easily more than a minute longer than the conventional system's transit time. Second, the propeller rpm would change continuously, although smoothly. After considering these shortcomings, I decide the linear approach profile is inefficient.

At this point, I've become quite familiar with the ferryboat control problem through doing all the

simulations. I know what the control surface should look like for the system to behave the way I want. I'm ready to use a nonlinear membership set for the fuzzy variable Distance. In this case, I've established that the profile created by the conventional control system (**Fig 2**) is a good guide, at least for the problem as I have posed it.

I need a fuzzy set for Distance for which set membership is proportional to the square root of the distance where 200m = 1. I determine the intermediate points from the equation

$$\text{membership} = \sqrt{\frac{\text{distance}}{200}}.$$

Fig 9 shows the nonlinear Positive_Large membership set for Distance.

One last modification is to add gain to tighten the control surface around the approach profile (the 0-rpm line). Without increased gain, the boat could deviate significantly from the desired approach profile and could crash into the dock before it has been slowed sufficiently. Of the several ways I could increase the gain, I chose to redefine the fuzzy sets for RPM_Command as **Fig 10** shows. The positive and negative rpm commands now exceed the possible propeller rpm by a factor of 10. A bounding function limits the output rpm command to the physical system limits of ±1000 rpm.

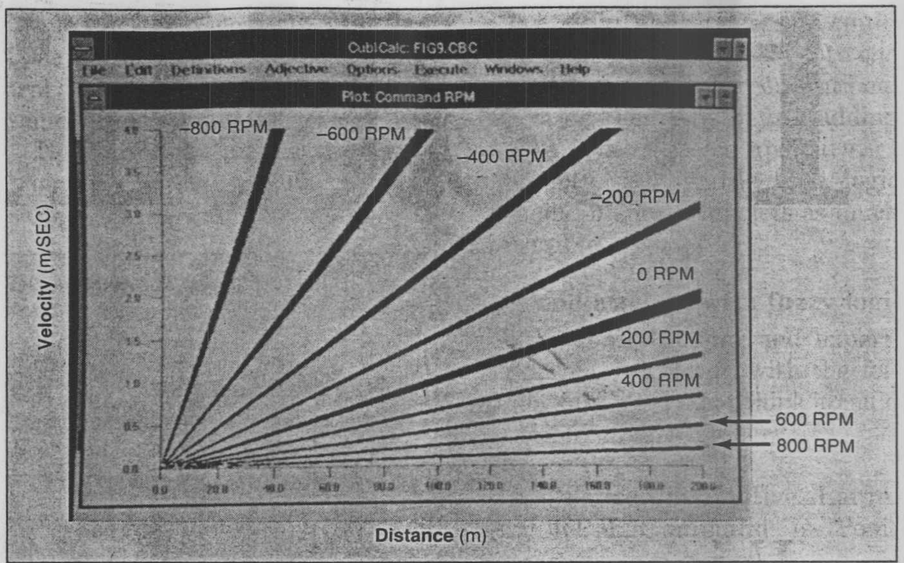

Fig 8—My second attempt at a fuzzy-logic control system results in this control surface. The linear velocity-versus-distance profiles are perhaps an improvement, but the system is still not acceptable for ferryboat docking.

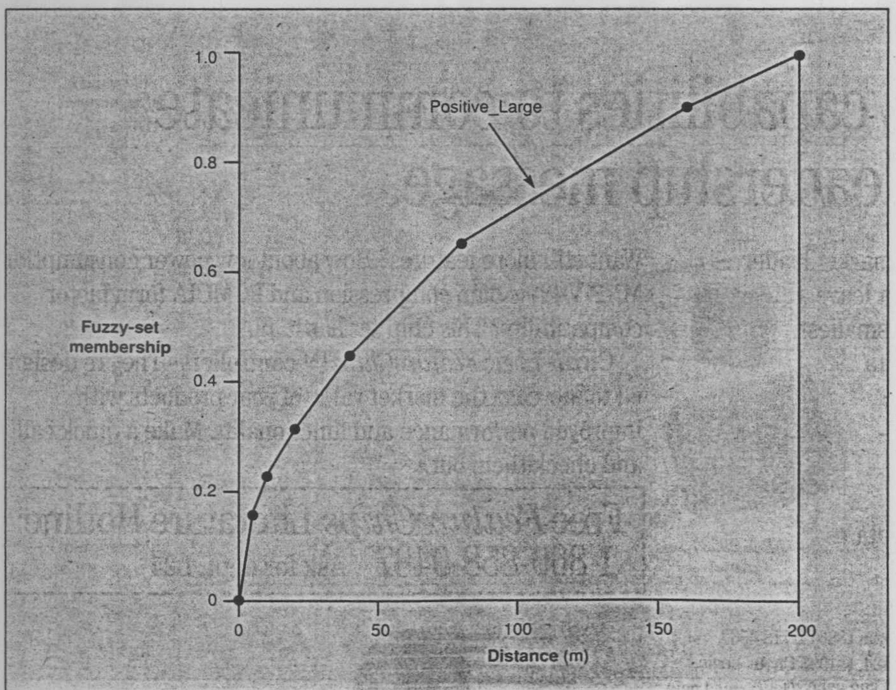

Fig 9—Nonlinear membership sets let you fine-tune fuzzy-logic systems. Here, I altered the Distance set membership to provide the desired distance-versus-velocity profile.

Fig 11 shows the control surface for this third fuzzy-logic system. It is essentially similar to the control surface of the conventional control system in **Fig 2**. I used only six intermediate points to create the nonlinear Distance membership curve. Thus, the control surface is not completely smooth, but you could make it as smooth as you needed by adding more points to the Distance membership curve.

Applying fuzzy logic

At this point I've accomplished my mission and learned a lot about using fuzzy logic. Although I still have lots more to learn, I can offer some comments for those trying to understand fuzzy logic or about to use it for the first time.

First, let's look at the example of the ferryboat control system. Conventional control methods work well, and the processing time would probably be the same whether a microcontroller was computing the

Designing a fuzzy-logic control system

conventional control equations or evaluating the fuzzy-logic rules. The relatively idealized ferryboat example probably wouldn't benefit from fuzzy logic, but making minor changes to the problem's definition might change the situation to favor fuzzy logic.

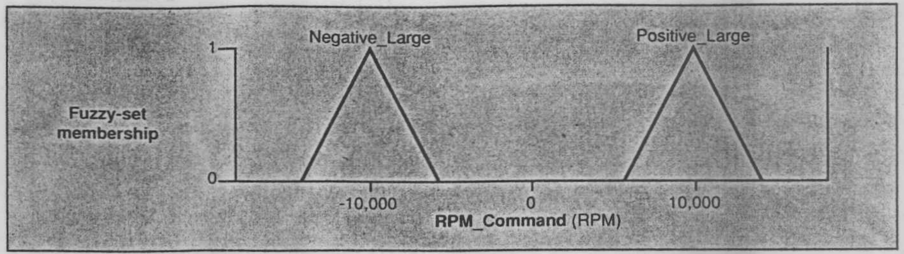

Fig 10—To increase the gain of the Fig 9 system, I altered the fuzzy sets for the RPM_Command output.

Consider the velocity-versus-distance profile. Perhaps for safety or other reasons the docking problem is redefined. Suppose that instead of constant deceleration, you wanted a gradually diminishing force that reached the dock at some small reverse-thrust setting. Now, the conventional control system requires a new equation or set of equations to create the new approach profile. My final fuzzy-logic system can accommodate *any* velocity-versus-distance profile; all I have to do is change the membership set for the Distance variable.

My experience probably holds true for control systems in general: If you can define the system using conventional control equations, fuzzy logic isn't necessary. If you have lots of exceptions to the equations, nonlinearities to accommodate, or no equations at all, fuzzy logic will be beneficial.

Don't look at fuzzy logic as a lazy way out of finding or deriving the proper equations for a control problem. You can attack just about any control system using fuzzy logic, but it's not always the best approach. Fuzzy logic isn't magic. To design a fuzzy-logic control system, you create the rules and the membership sets. You, the designer, provide all the knowledge to make the system work. Before completing the design, you will have acquired a thorough knowledge of the system.

That knowledge could come from equations, rules you create or obtain from an expert, trial-and-error experiments (fuzzy hacking), or knowing how the system's control surface should look. You should strive to acquire this knowledge as efficiently as possible. Equations are concentrated knowledge, and you should use them for fuzzy systems whenever possible. The ad-

Manufacturers of fuzzy-logic products

For more information on fuzzy-logic products such as those described in this article, circle the appropriate numbers on the Information Retrieval Service card or use EDN's Express Request service. When you contact any of the following manufacturers directly, please let them know you saw their products in EDN.

American Neuralogix Inc
411 Central Park Dr
Sanford, FL 32771
(407) 322-5608
FAX (407) 322-5609
Circle No. 302

Aptronix Inc
2150 N First St, Suite 300
San Jose, CA 95131
(408) 428-1888
FAX (408) 428-1884
Fuzzynet BBS (408) 428-1883 (8,N,1)
Circle No. 303

Byte Craft Ltd
421 King St N
Waterloo, Ontario, N2J 4E4 Canada
(519) 888-6911
FAX (519) 746-6751
Circle No. 304

Fuzzy Systems Engineering
Box 27390
San Diego, CA 27390
(619) 748-7384
Circle No. 305

Hitachi America Ltd
2000 Sierra Point Pkwy
Brisbane, CA 94005
(415) 589-8300
FAX (415) 583-4207
Circle No. 306

Hyperlogic Corp
1855 E Valley Pkwy, Suite 210
Escondido, CA 92027
(619) 746-2765
FAX (619) 746-4089
Circle No. 307

Inform Software Corp
1840 Oak Ave
Evanston, IL 60201
(708) 866-1838
FAX (708) 866-1808
Circle No. 308

Integrated Systems Inc
3260 Jay St
Santa Clara, CA 95054
(408) 980-1500
FAX (408) 980-0400
Circle No. 309

The Metus Systems Group
1 Griggs Lane
Chappaqua, NY 10514
(914) 238-0647
FAX (914) 238-0837
Circle No. 310

Modico Inc
Box 8485
Knoxville, TN 37996
(615) 531-7008
FAX (615) 693-6645
Circle No. 311

Motorola Inc
Box 1466
Austin, TX 78767
(512) 891-2840
Freeware BBS (512) 891-3733
Circle No. 312

Omron Electronics Inc
1 E Commerce Dr
Schaumburg, IL 60173
(708) 843-7900
FAX (708) 843-7787
Circle No. 313

Togai Infralogic Inc
5 Vanderbilt
Irvine, CA 92718
(714) 975-8522
FAX (714) 975-8524
Circle No. 314

VOTE . . .
Please also use the Information Retrieval Service card to rate this article (circle one):
High Interest 491 Medium Interest 492 Low Interest 493

Designing a fuzzy-logic control system

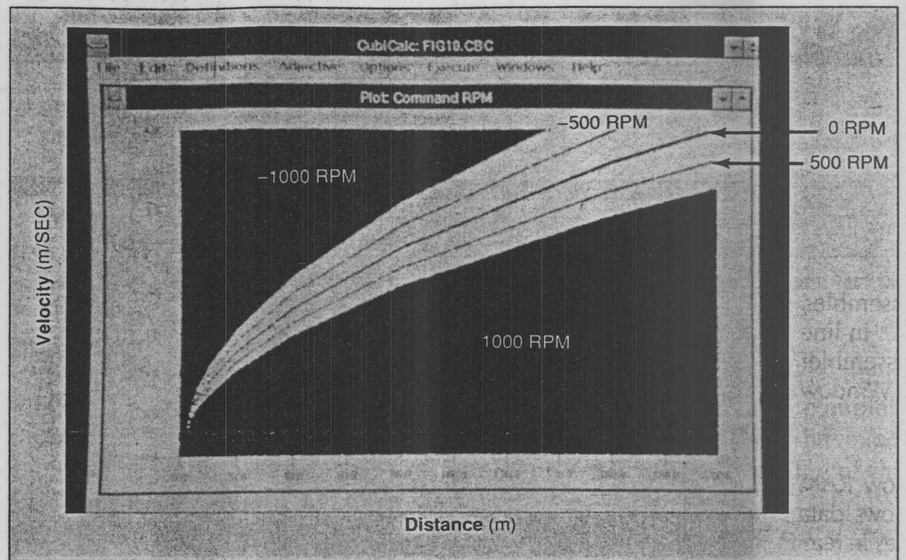

Fig 11—My third attempt at a fuzzy-logic ferryboat-docking control system yields an acceptable solution, as you can see from the smooth control surface.

vantage of using fuzzy logic instead of a conventional approach is that equations aren't absolutely necessary—you aren't constrained by what you can describe with equations.

In the ferryboat example, knowing that a velocity proportional to the square root of the distance will use constant thrust and result in the shortest docking time is useful. If you didn't have that knowledge to start with, you'd eventually discover it by trial and error. Just because you're using fuzzy logic on a problem doesn't mean you can't use your mathematical understanding of the problem. In other words, even though you may consider a problem too difficult to approach with conventional control equations, any mathematical understanding you have of the problem will prove useful.

Probably the most difficult part of learning to use fuzzy logic is rethinking the way you look at control problems. Without even realizing it, many engineers define control-system requirements to fit the methods they will use later to create a control system. Those without an understanding of fuzzy logic tend to think of control systems as combinations of Boolean logic statements for making decisions and conventional control equations for continuously variable functions. Fuzzy logic spans the gap between the two. EDN

References

1. Brubaker, David I, "Fuzzy-logic basics: Intuitive rules replace complex math," *EDN*, June 18, 1992, pg 111.

2. Brubaker, David I, and Cedric Sheerer, "Fuzzy-logic system solves complex control problem," *EDN*, June 18, 1992, pg 121.

3. Legg, Gary, "Special tools and chips make fuzzy logic simple," *EDN*, July 6, 1992, pg 68.

Article Interest Quotient
(Circle One)
High 491 Medium 492 Low 493